模型评价与模型改善

CS245　数据科学基础　陆朝俊
叶泽林 515030910468

1　问题描述

2　解决方案[1]

2.1　数据集的建立及模型的选择

2.2　探索模型单独工作的性能

1. 直接划分训练集和测试集

   无可奉告

2. 交叉验证

3. 网格搜索

2.3　探索模型组合工作的性能

3　结果展示

3.1　模型单独工作的性能

3.1.1　直接划分训练集和测试集

3.1.2　交叉验证

3.1.3　网格搜索

3.2　模型组合工作的性能

3.2.1　bagging

3.2.2　boosting

[1]本次作业的所有代码实现可参见附录 A.3

A 附录

A.1 Boston 数据集特征信息

表 1: Boston 数据集特征信息

编号	特征名	特征含义
1	CRIM	城镇人均犯罪率
2	ZN	住宅用地超过 25000 sq.ft. 的比例
3	INDUS	城镇非零售商用土地的比例
4	CHAS	查理斯河空变量（如果边界是河流，则为 1；否则为 0）
5	NOX	一氧化氮浓度
6	RM	住宅平均房间数
7	AGE	1940 年之前建成的自用房屋比例
8	DIS	到波士顿五个中心区域的加权距离
9	RAD	辐射性公路的接近指数
10	TAX	每 10000 美元的全值财产税率
11	PTRATIO	城镇师生比例
12	B	1000(Bk-0.63)2，其中 Bk 指代城镇中黑人的比例
13	LSTAT	人口中地位低下者的比例

A.2 PCA 算法在主成分个数为 3 时的进一步可视化

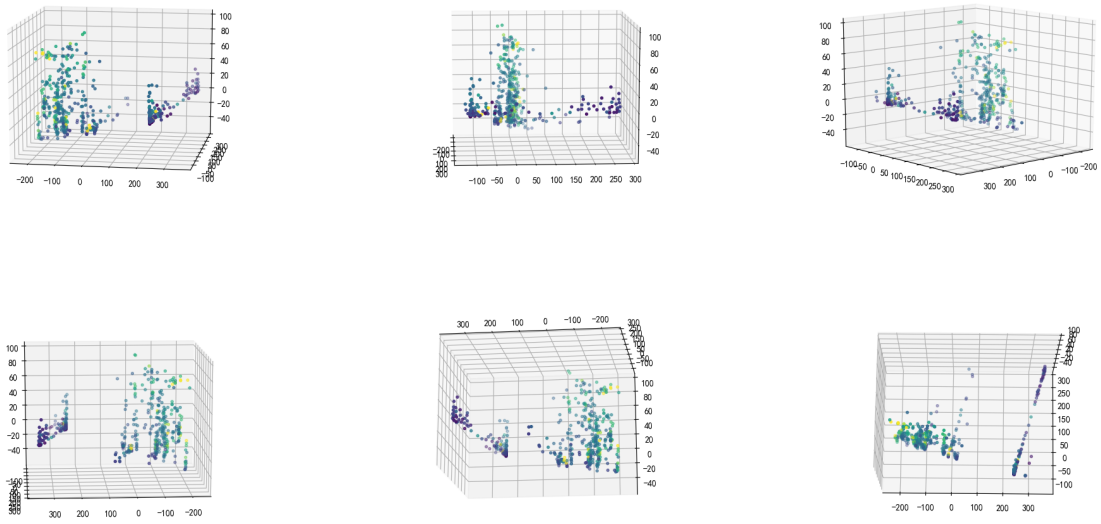

图 1: PCA 效果进一步可视化（主成分个数为 3 时）

A.3 主要代码

```
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']     # display Chinese
plt.rcParams['axes.unicode_minus'] = False       # display minus sign
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA

import os
import time

class bostonAnalyzer(object):
    def __init__(self):
        # load dataset
        self.dataset = datasets.load_boston()
        self.data = self.dataset.data
        self.target = self.dataset.target

        # for plot
        self.var = []
        self.t = []

        if not os.path.exists('report/img'):
            os.makedirs('report/img')

    def run(self, vis=False):
        '''
        Run PCA for 1-13 dimensions
        :param vis: True: plot dim 1-3; False: not plot
        '''

        self.var.clear()
        self.t.clear()

        with open('report/result.txt', 'w') as f:
            for i in range(13):
                t = time.time()
                pca_op = PCA(n_components=i)
                pca_res = pca_op.fit_transform(self.data)
                t = time.time() - t

                self.var.append(pca_op.explained_variance_ratio_.sum()
                    )
                self.t.append(t)

                # write log
                f.write('###### Dimension %d ######\n' % i)
                f.write(str(pca_res.shape) + '\n')
```

```
47                  f.write(str(pca_op.explained_variance_ratio_) + '\n')
48                  f.write(str(pca_op.explained_variance_ratio_.sum()) +
                        '\n')
49                  f.write(str(t) + '\n\n')
50
51                  # visualize for debug & plot
52                  if vis:
53                      if i == 1:
54                          # plot 1 dimension
55                          plt.scatter(pca_res[:, 0], pca_res[:, 0], s
                                =14, c=self.target)
56                          plt.savefig('report/img/pca-%d' % i)
57                          plt.show()
58                      elif i == 2:
59                          # plot 2 dimensions
60                          plt.scatter(pca_res[:,0], pca_res[:,1], s=8, c
                                =self.target)
61                          plt.savefig('report/img/pca-%d' % i)
62                          plt.show()
63                      elif i == 3:
64                          # plot 3 dimensions
65                          ax = plt.subplot(projection='3d')
66                          ax.scatter(pca_res[:, 0], pca_res[:, 1],
                                pca_res[:, 2], s=8, c=self.target)
67                          plt.savefig('report/img/pca-%d' % i)
68                          plt.show()
69
70      def show(self):
71          '''
72          Show the basic info of Boston dataset & plot k-lines
73          '''
74          print(self.data.shape)
75          print(self.target.shape)
76          self._k_line_radio()
77          self._k_line_time()
78
79      def _k_line_radio(self):
80          '''
81          Plot k-line of variant radio
82          '''
83          x = list(range(len(self.var)))
84          plt.scatter(x, self.var, s=14, c='r')
85          plt.plot(x, self.var)
86          plt.xlabel('主成分个数')
87          plt.ylabel('降维后各特征方差比例之和')
88          plt.savefig('report/img/kline-radio')
89          plt.show()
90
```

```
91      def _k_line_time(self):
92          '''
93          Plot k-line of time
94          '''
95          x = list(range(len(self.t)))
96          plt.scatter(x, self.t, s=14, c='r')
97          plt.plot(x, self.t)
98          plt.xlabel('主成分个数')
99          plt.ylabel('PCA算法消耗时间(s)')
100         plt.savefig('report/img/kline-time')
101         plt.show()
102
103 if __name__ == '__main__':
104     bA = bostonAnalyzer()
105     bA.run(True)
106     bA.show()
```